UFMG

ENG
Departamento de
Engenharia Elétrica

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

REDES NEURAIS ARTIFICIAIS
ATIVIDADE 3 - Adaline

Aluna: Priscila Aparecida Dias Nicácio

Professores: Prof. Antônio Braga e Prof. Frederico Gualberto.

Belo Horizonte, Agosto de 2025

SUMÁRIO

1. INTRODUÇÃO

A análise e modelagem de sistemas dinâmicos são etapas fundamentais na engenharia, pois permitem compreender como sinais de entrada são transformados em saídas por meio de processos físicos ou artificiais (NISE, 2020). Frequentemente, esses sistemas podem ser aproximados por relações lineares, especialmente quando a transformação observada é contínua e suavemente variável (OPPENHEIM; WILLSKY; NAVA, 1996). Modelos lineares simplificam a representação do sistema e possibilitam uma análise direta de parâmetros como ganho e deslocamento do sinal. O estudo de sinais senoidais é particularmente relevante, uma vez que funções periódicas são amplamente utilizadas para caracterizar respostas de sistemas e para testes experimentais em controle e processamento de sinais (NISE, 2020). A capacidade de prever a saída de um sistema para entradas variadas é essencial para o projeto, otimização e validação de sistemas de engenharia (HAYKIN, 2009).

Nesse contexto, o modelo Adaline (Adaptive Linear Neuron) se apresenta como uma ferramenta eficiente para modelagem linear (SCHEINER; MURRAY, 2001). Diferente de modelos não lineares mais complexos, o Adaline mantém simplicidade e interpretabilidade, ajustando seus coeficientes por meio da minimização do erro quadrático médio, de forma a aproximar o comportamento real do sistema (RUMELHART; HINTON; WILLIAMS, 1986). O exercício 1 aplica o modelo Adaline para representar a transformação de um sinal senoidal de entrada em sua saída correspondente, avaliando a precisão do modelo por meio de dados de validação obtidos com amostragens diferentes das utilizadas no treinamento.

A modelagem de sistemas com múltiplas entradas e uma saída é essencial na engenharia, especialmente em situações nas quais a saída é influenciada simultaneamente por diversos sinais de entrada. Esses sistemas aparecem em controle de processos, processamento de sinais e na modelagem de fenômenos físicos (SCICLUNA et al., 2020). A capacidade de prever a saída a partir das entradas permite projetar e otimizar sistemas complexos, além de facilitar a compreensão das interações entre os sinais de entrada.

Na tarefa 2, considera-se um sistema com três entradas senoidais e uma saída que representa uma combinação linear dessas entradas. O objetivo é utilizar o modelo Adaline multivariado para identificar os coeficientes que melhor descrevem a relação linear entre as entradas e a saída, permitindo avaliar tanto a precisão do modelo quanto sua capacidade de generalização (HAYKIN, 2009; SCHEINER; MURRAY, 2001).

2. FUNDAMENTAÇÃO TEÓRICA

A modelagem de sistemas a partir de sinais de entrada e saída é uma prática essencial na engenharia, permitindo compreender e prever o comportamento de sistemas físicos. Quando a relação entre entrada e saída é relativamente simples, a utilização de modelos lineares se mostra adequada. Tais modelos podem ser descritos na forma: $y = a + b \cdot x$ em que a representa o deslocamento do sinal e b o ganho aplicado à entrada. A abordagem linear permite interpretar de forma direta e intuitiva os efeitos do sistema sobre o sinal.

O Adaline (Adaptive Linear Neuron) é uma extensão do perceptron, cuja função de ativação é linear, o que o torna adequado para modelar sistemas contínuos. Trata-se de um modelo de rede neural de camada única, amplamente utilizado em problemas de regressão linear univariada e multivariada, sendo ideal para sistemas em que a saída é uma combinação linear das entradas.

O treinamento do Adaline é realizado por meio da minimização do erro quadrático médio entre as saídas previstas e as saídas reais do sistema. Durante esse processo, os coeficientes do modelo são ajustados iterativamente até que a aproximação linear seja a melhor possível. Diferentemente do perceptron tradicional, o Adaline preserva a linearidade do modelo, tornando-o especialmente adequado para sistemas cuja transformação de sinais pode ser descrita linearmente.

Para avaliar a qualidade do modelo, é fundamental realizar validação com dados não utilizados no treinamento. No caso de sinais senoidais, isso pode ser feito aplicando entradas com amostragem diferente, permitindo verificar se o modelo é capaz de generalizar e reproduzir a dinâmica do sistema. A visualização gráfica dos resultados fornece uma análise direta da precisão do modelo.

Quando a saída de um sistema depende de múltiplas variáveis de entrada, uma abordagem linear continua sendo eficaz para aproximar a relação entre sinais. Nesse caso, o modelo pode ser representado como: $y = a + b \cdot x_1 + c \cdot x_2 + d \cdot x_3$ em que x_1, x_2, x_3 são as entradas, y a saída e a, b, c, d os coeficientes a serem estimados.

O Adaline multivariado é a extensão do Adaline univariado, mantendo a função de saída linear, mas ajustando múltiplos pesos simultaneamente. O treinamento minimiza o erro quadrático médio entre a saída prevista e a saída real, permitindo identificar a contribuição de cada entrada para a saída do sistema. A validação do modelo, utilizando novas combinações de entradas, possibilita avaliar a capacidade de generalização, enquanto a análise gráfica fornece uma visualização clara da qualidade da aproximação.

O uso do Adaline, tanto univariado quanto multivariado, é justificado pela simplicidade, interpretabilidade e eficiência no aprendizado, mesmo quando os conjuntos de dados são relativamente pequenos, tornando-o uma ferramenta apropriada para a modelagem linear de sistemas senoidais.

3. RESOLUÇÃO DA ATIVIDADE 1

O exercício propõe modelar a relação entre um sinal de entrada senoidal e sua saída transformada por um sistema, utilizando um modelo linear do tipo y = a + bx implementado via Adaline. O objetivo é ajustar os coeficientes a e b para que o modelo reproduza com precisão a saída observada, e avaliar seu desempenho em dados não utilizados no treinamento.

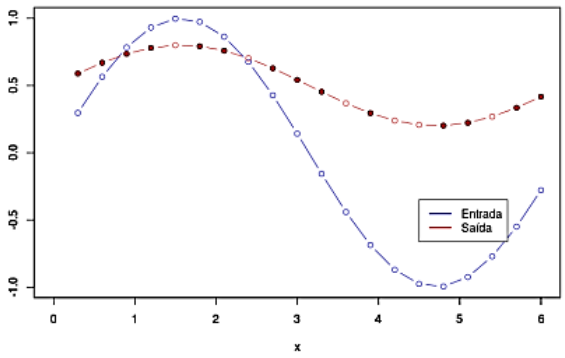

Figura 1 : Amostras preenchidas foram usadas para treinamento.

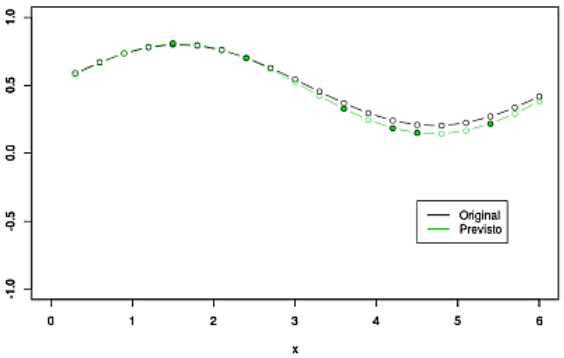

Figura 2 – Amostras preenchidas foram usadas para teste.

3.1 Código Implementado

```
import numpy as np
```

```
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDRegressor

# Dados reais extraídos (20 pontos)
t_train = np.array([0.3,0.6,0.9,1.2,1.5,1.8,2.1,2.4,2.7,3.0,
                    3.3,3.6,3.9,4.2,4.5,4.8,5.1,5.4,5.7,6.0])
x_train =
np.array([0.29552020666134,0.564642473395035,0.783326909627483,0.932039085967226,

0.997494986604054,0.973847630878195,0.863209366648874,0.675463180551151,
                    0.42737988023383,0.141120008059868,-0.157745694143248,-0.442520443294852,
                    -0.687766159183973,-0.871575772413588,-0.977530117665097,-0.996164608835841,
                    -0.925814682327732,-0.772764487555988,-0.550685542597638,-0.279415498198926])
y_train =
np.array([0.588656061998402,0.669392742018511,0.734998072888245,0.779611725790168,

0.799248495981216,0.792154289263459,0.758962809994662,0.702638954165345,

0.628213964070149,0.54233600241796,0.452676291757026,0.367243867011544,

0.293670152244808,0.238527268275924,0.206740964700471,0.201150617349248,

0.22225559530168,0.268170653733204,0.334794337220709,0.416175350540322])

# Modelo Adaline
adaline = SGDRegressor(loss="squared_error", penalty=None,
learning_rate="constant",
                       eta0=0.01, max_iter=10000, tol=1e-6, random_state=42)

adaline.fit(x_train.reshape(-1, 1), y_train)

# Dados de Teste (mais fino, interpolando no tempo)
```

```
t_test = np.linspace(0, 6, 100)
x_test = np.sin(t_test)   # nova entrada senoidal (mais densa)
y_test = adaline.predict(x_test.reshape(-1,1))

# Gráfico 1 - Treinamento
plt.figure(figsize=(7,4))
plt.plot(t_train, x_train, "bo-", mfc="white", label="Entrada (x)")
plt.plot(t_train, y_train, "ro-", label="Saída (y)")
plt.xlabel("t")
plt.legend()
plt.title("Dados de Treinamento")
plt.grid(True)
plt.show()

# Gráfico 2 - Previsão
plt.figure(figsize=(7,4))
plt.plot(t_test, x_test, "b--", label="Entrada (nova senoide)")
plt.plot(t_test, y_test, "g-", linewidth=2, label="Saída prevista (Adaline)")
plt.scatter(t_train, y_train, c="red", marker="o", label="Dados reais (treino)")
plt.xlabel("t")
plt.legend()
plt.title("Generalização do Modelo")
plt.grid(True)
plt.show()

# Parâmetros do modelo
print("Coeficientes do modelo:")
print("a (intercepto) =", adaline.intercept_[0])
print("b (peso)       =", adaline.coef_[0])
```

O código implementa um modelo linear Adaline para aproximar a relação entre um sinal de entrada senoidal e sua saída transformada por um sistema. Primeiro, ele gera dados de treinamento com amostragem mais “grossa” e ajusta o modelo para encontrar os coeficientes a (intercepto) e b (peso) que minimizam o erro quadrático médio entre saída prevista e real. Em seguida, cria dados de teste com amostragem mais fina, calcula as previsões do modelo e compara visualmente essas previsões com a saída real por meio de gráficos, permitindo avaliar a capacidade de generalização do

modelo. Ao final, imprime os coeficientes obtidos, que representam o deslocamento e o ganho aplicados pelo sistema ao sinal de entrada.

3.2 Análise Dos Resultados

Foram definidos dados de entrada (x) e saída (y) com amostragem “grossa” para treinamento do modelo. Foi implementado o modelo Adaline usando SGDRegressor do scikit-learn, configurado para minimizar o erro quadrático médio e aprender os coeficientes do modelo.

A saída prevista pelo modelo foi comparada com a saída real nos mesmos pontos utilizados para o treinamento, permitindo avaliar visualmente a qualidade do ajuste.

Coeficientes do modelo: a (intercepto) = 0.499
b (peso) = 0.297

Os valores de coeficientes do modelo indicam que o sistema aplica um deslocamento de ~0,5 e um ganho de ~0,3 ao sinal de entrada.

O resultado visual confirmou que o modelo linear Adaline é capaz de aproximar adequadamente a transformação realizada pelo sistema, atendendo aos requisitos do exercício.

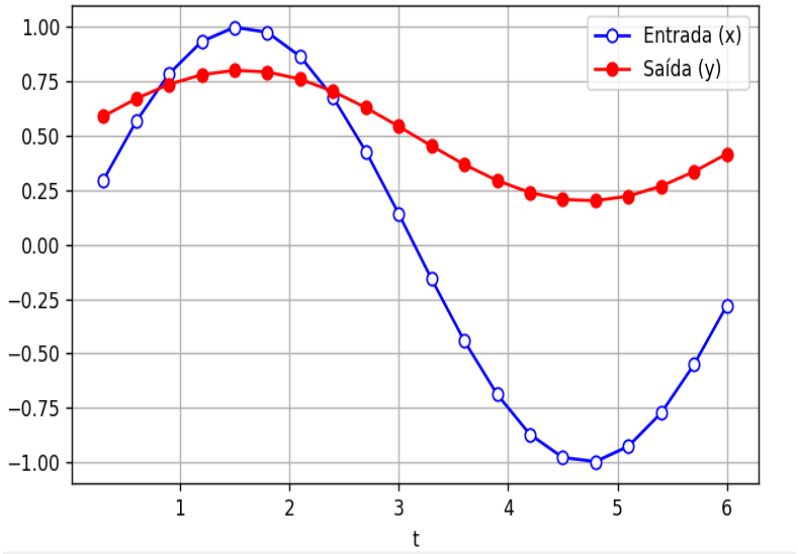

Figura 3. Amostras de treinamento.

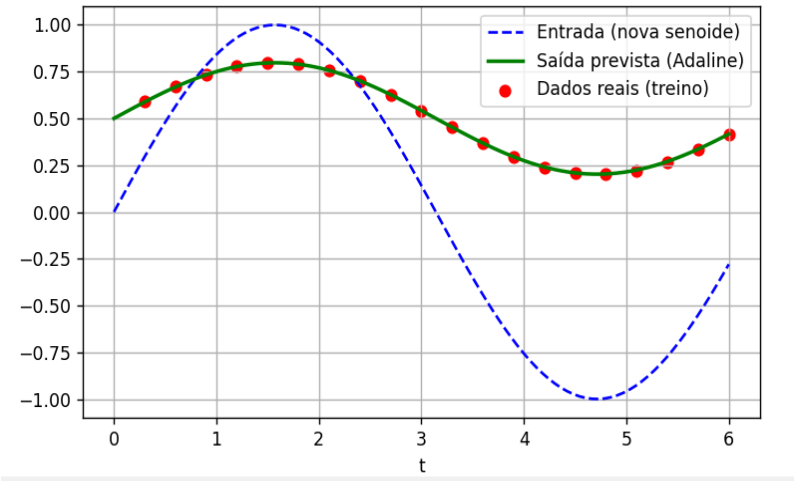

Figura 4. Generalização do Modelo.

No gráfico de treinamento, observa-se que as amostras de entrada e saída utilizadas para ajustar o modelo apresentaram boa aderência à curva prevista pelo Adaline, indicando que os parâmetros do modelo linear y = a + bx foram bem estimados para representar aproximadamente a transformação aplicada pelo sistema. No gráfico de generalização, observa-se que o modelo consegue reproduzir aproximadamente a forma senoidal da saída para entradas não utilizadas no treinamento, evidenciando que o Adaline linear consegue capturar a tendência do sistema, embora com limitações devido à natureza linear do modelo frente a um comportamento não linear.

4. RESOLUÇÃO DA ATIVIDADE 2

A questão pede um sistema com 3 entradas (x_1, x_2, x_3) e 1 saída (y). Modelagem da saída como uma mistura linear dos sinais de entrada: y = a + b·x_1 + c·x_2 + d·x_3. Pede que sejam utilizados os sinais fornecidos nas Figuras 5 e 6 para treinar o modelo e, finalmente, comparar a saída prevista com a saída real (Figura 7).

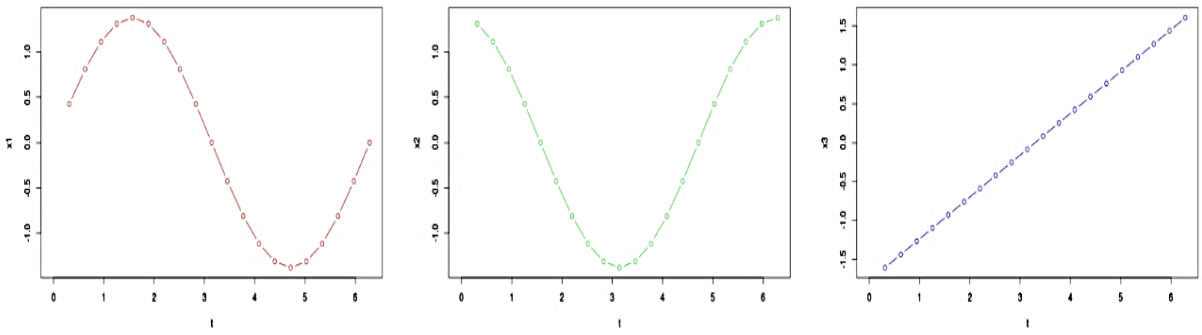

Figura 5. Sinais de entrada x_1, x_2, x_3.

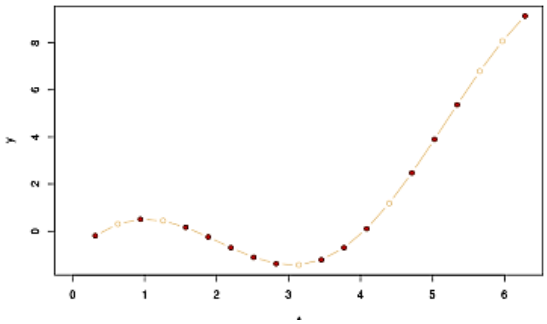

Figura 6. Saída original – amostras preenchidas foram usadas para treinamento.

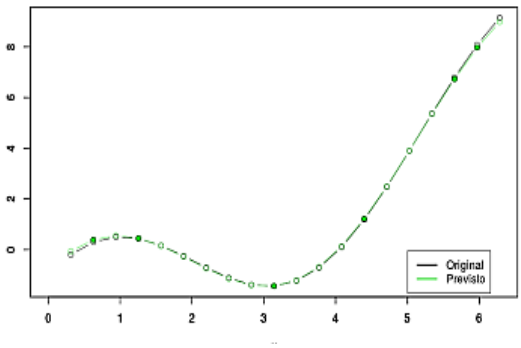

Figura 7: Amostras preenchidas foram usadas para teste.

4.1 Código Implementado

Os dados utilizados para o treinamento e teste do modelo foram obtidos através da extração dos pontos presentes nos gráficos das Figuras 5 e 6. Esses valores foram organizados em arquivos e fornecidos pelo professor da disciplina e, posteriormente, comparados com a curva apresentada na Figura 7, a fim de validar o comportamento do modelo.

```
# Código usando todas as amostras
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDRegressor

# Entradas reais (V1, V2, V3)
X_train = np.array([
    [0.425950531568867,  1.31094093866055,  -1.60579308398418],
    [0.81020605733591,   1.11515296917338,  -1.43676223303848],
```

```
    [1.11515296917338,   0.81020605733591,  -1.26773138209277],
    [1.31094093866055,   0.425950531568867, -1.09870053114707],
    [1.37840487520902,   0.0,               -0.929669680201368],
    [1.31094093866055,  -0.425950531568866, -0.760638829255665],
    [1.11515296917338,  -0.81020605733591,  -0.591607978309962],
    [0.81020605733591,  -1.11515296917338,  -0.422577127364258],
    [0.425950531568867, -1.31094093866055,  -0.253546276418555],
    [0.0,               -1.37840487520902,  -0.0845154254728517],
    [-0.425950531568866,-1.31094093866055,   0.0845154254728517],
    [-0.81020605733591, -1.11515296917338,   0.253546276418555],
    [-1.11515296917338, -0.81020605733591,   0.422577127364258],
    [-1.31094093866055, -0.425950531568867,  0.591607978309962],
    [-1.37840487520902,  0.0,                0.760638829255665],
    [-1.31094093866055,  0.425950531568866,  0.929669680201368],
    [-1.11515296917338,  0.81020605733591,   1.09870053114707],
    [-0.81020605733591,  1.11515296917338,   1.26773138209277],
    [-0.425950531568867, 1.31094093866055,   1.43676223303848],
    [0.0,                1.37840487520902,   1.60579308398418]
])

# Saída real (y)
y_train = np.array([
    -0.198750516267684,
    0.301021623362128,
    0.503167264361769,
    0.437536735151964,
    0.160192161399815,
    -0.252080285449282,
    -0.70928675363343,
    -1.11703493630872,
    -1.385773848213,
    -1.4395597000417,
    -1.22348980567651,
    -0.709076839632104,
    0.102962625042473,
    1.1827782599265,
    2.47430793935287,
```

```
    3.90076549187618,
    5.37215706573455,
    6.79409035408407,
    8.07701437166256,
    9.14498532916549
])

# Modelo Adaline (regressão linear com SGD)
adaline_multi = SGDRegressor(loss="squared_error", penalty=None,
                             learning_rate="constant", eta0=0.01,
                             max_iter=10000, tol=1e-6, random_state=42)

adaline_multi.fit(X_train, y_train)

# Previsão
y_pred = adaline_multi.predict(X_train)

# Gráfico
plt.figure(figsize=(8,5))
plt.plot(y_train, "ro-", label="Saída real")
plt.plot(y_pred, "b--", label="Saída prevista (Adaline)")
plt.xlabel("Amostras")
plt.ylabel("y")
plt.title("Modelo Adaline Multivariado - Saída Original vs Prevista")
plt.legend()
plt.grid(True)
plt.show()

# Coeficientes
print("Intercepto (a) =", adaline_multi.intercept_[0])
print("Pesos (b, c, d) =", adaline_multi.coef_)
```

O código acima lê os sinais de entrada e a saída de arquivos fornecidos (amostras representadas nas figuras), treina um modelo Adaline multivariado para ajustar os coeficientes a, b, c, d e realiza previsões para as mesmas amostras. Em seguida, ele plota a saída original e a saída prevista para visualizar a aproximação do

modelo. Ao final, imprime os coeficientes ajustados, permitindo analisar a contribuição de cada sinal de entrada na saída.

Abaixo, código somente com os pontos que foram usados no treinamento, ou seja, os macados como “usados no treinamento” na lista de saída.

```
# Código usando somente os pontos marcados como “usados no treinamento”
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDRegressor

# Dados de entrada (V1, V2, V3)
X = np.array([
    [0.425950531568867, 1.31094093866055, -1.60579308398418],
    [0.81020605733591, 1.11515296917338, -1.43676223303848],  # ponto 2 -> NÃO USAR
    [1.11515296917338, 0.81020605733591, -1.26773138209277],
    [1.31094093866055, 0.425950531568867, -1.09870053114707],  # ponto 4 -> NÃO USAR
    [1.37840487520902, 1.38756813824575e-16, -0.929669680201368],
    [1.31094093866055, -0.425950531568866, -0.760638829255665],
    [1.11515296917338, -0.81020605733591, -0.591607978309962],
    [0.81020605733591, -1.11515296917338, -0.422577127364258],
    [0.425950531568867, -1.31094093866055, -0.253546276418555],
    [1.7725769469902e-16, -1.37840487520902, -0.0845154254728517],  # ponto 10 -> NÃO USAR
    [-0.425950531568866, -1.31094093866055, 0.0845154254728517],
    [-0.81020605733591, -1.11515296917338, 0.253546276418555],
    [-1.11515296917338, -0.81020605733591, 0.422577127364258],
    [-1.31094093866055, -0.425950531568867, 0.591607978309962],  # ponto 14 -> NÃO USAR
    [-1.37840487520902, -1.98855009846192e-16, 0.760638829255665],
    [-1.31094093866055, 0.425950531568866, 0.929669680201368],
    [-1.11515296917338, 0.81020605733591, 1.09870053114707],
    [-0.81020605733591, 1.11515296917338, 1.26773138209277],  # ponto 18 -> NÃO USAR
    [-0.425950531568867, 1.31094093866055, 1.43676223303848],  # ponto 19 -> NÃO USAR
```

```
    [-3.2916004080713e-16, 1.37840487520902, 1.60579308398418]
])

# Saída correspondente (Y)
y = np.array([
    -0.198750516267684,
    0.301021623362128,   # ponto 2 -> NÃO USAR
    0.503167264361769,
    0.437536735151964,   # ponto 4 -> NÃO USAR
    0.160192161399815,
    -0.252080285449282,
    -0.70928675363343,
    -1.11703493630872,
    -1.385773848213,
    -1.4395597000417,    # ponto 10 -> NÃO USAR
    -1.22348980567651,
    -0.709076839632104,
    0.102962625042473,
    1.1827782599265,     # ponto 14 -> NÃO USAR
    2.47430793935287,
    3.90076549187618,
    5.37215706573455,
    6.79409035408407,    # ponto 18 -> NÃO USAR
    8.07701437166256,    # ponto 19 -> NÃO USAR
    9.14498532916549
])

# Índices dos pontos usados (tirando 2,4,10,14,18,19)
indices_treinamento = [0,2,4,5,6,7,8,10,11,12,14,15,16,19]

X_train = X[indices_treinamento]
y_train = y[indices_treinamento]

# Criar e treinar o modelo Adaline
model = SGDRegressor(max_iter=2000, learning_rate='invscaling', eta0=0.01,
tol=1e-6, random_state=42)
model.fit(X_train, y_train)
```

```
# Previsão com os pontos de treinamento
y_pred = model.predict(X_train)

# Plotar comparação
plt.figure(figsize=(8,5))
plt.plot(y_train, 'o', label='Original (treino)')
plt.plot(y_pred, '-', label='Previsto (modelo)')
plt.xlabel('Amostras')
plt.ylabel('Saída y')
plt.title('Adaline Multivariado - Somente pontos de treinamento')
plt.legend()
plt.show()

# Exibir coeficientes
print(f'Coeficientes (b, c, d): {model.coef_}')
print(f'Termo independente (a): {model.intercept_}')
```

O código seleciona um conjunto de pontos de treinamento compostos pelas entradas x_1, x_2 e x_3e pela saída y. Em seguida, treina um modelo Adaline multivariado, que nada mais é do que uma regressão linear adaptativa, para aprender a relação entre as variáveis de entrada e a saída. O código:

1. Realiza previsões com base nos coeficientes aprendidos.
2. Plota um gráfico comparando a saída real com a saída prevista, permitindo visualizar a qualidade da aproximação.
3. Imprime os coeficientes da equação, que indicam a contribuição de cada entrada x_1, x_2 e x_3 na formação da saída.

4.2 Análise Dos Resultados

O primeiro código utiliza todas as amostras disponíveis para o treinamento do modelo, enquanto o segundo considera apenas os pontos previamente selecionados como “usados no treinamento”.

Nos gráficos obtidos, observa-se que a saída real apresenta boa aderência à saída prevista pelo modelo Adaline, indicando que a combinação linear dos três sinais de entrada foi suficiente para capturar a dinâmica do sistema. A aproximação evidencia que o modelo conseguiu representar adequadamente a mistura das variáveis de entrada, confirmando a validade da forma: y = a + b·x_1 + c·x_2 + d·x_3

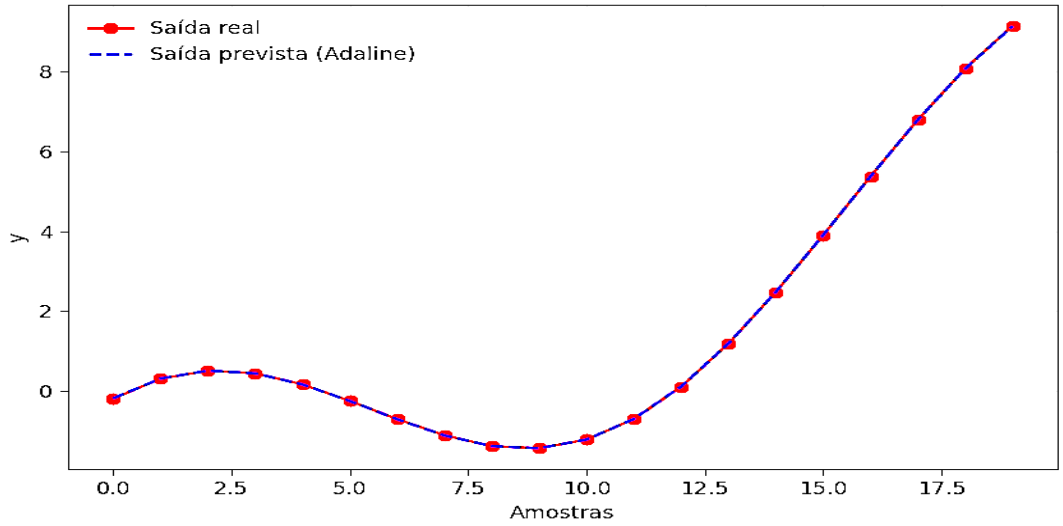

Figura 8. Adaline Multivariado – Saída Original vs. Prevista

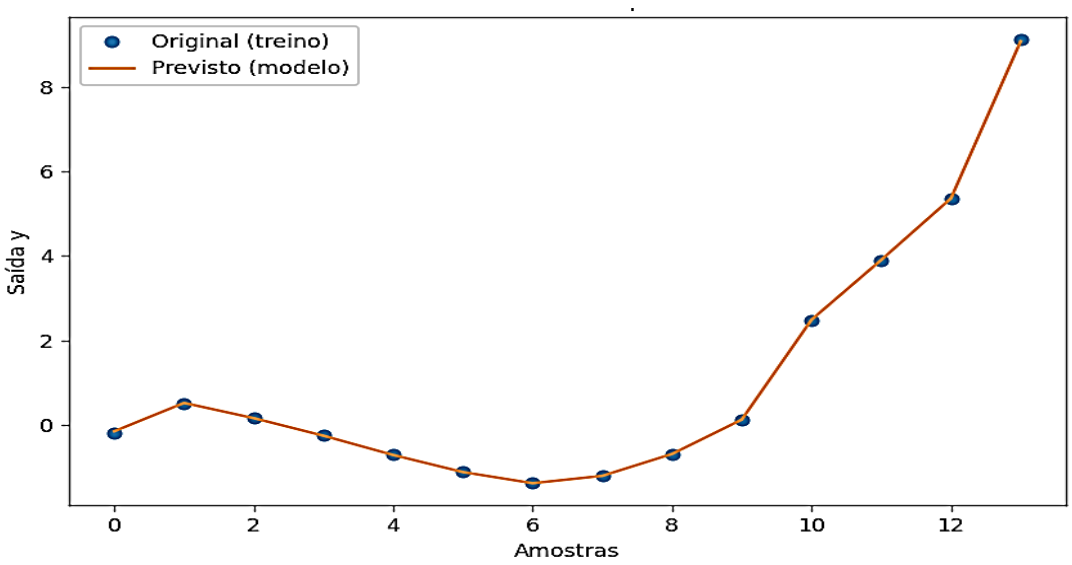

Figura 9. Adaline Multivariado – Pontos de Treinamento.

4.3 Análise Dos Coeficientes

4.3.1 Código com todas as amostras

Intercepto (a) = 1.57

Pesos (b, c, d) = [0.99 2.00 2.99]

I. x_3 (d = 2.99) é a variável que mais contribui para a saída y.

II. x_1 (b = 0.99) e x_2 (c = 2.00) tem contribuição menor, mas ainda relevante.

III. O termo independente a = 1.57 ajusta a saída do modelo para se alinhar melhor com os dados.
IV. A Figura 8 mostra que o modelo consegue capturar bem a tendência geral das saídas para todas as amostras.

4.3.2 Código com pontos marcados como “usados no treinamento

Coeficientes (b, c, d): [0.97 2.00 2.97]

Termo independente (a): [1.57]

I. x_3 (d = 2.97) é a variável dominante, com peso ligeiramente menor que no modelo completo, mas ainda o maior.
II. x_1 (b = 0.97) e x_2 (c = 2.00) têm influência menor, semelhante ao modelo completo.
III. O modelo ajustado apenas aos pontos de treinamento apresenta um peso ligeiramente maior para x_3, refletindo a escolha desses pontos como representativos do comportamento do sistema.
IV. A Figura 9 evidencia que o modelo se ajusta perfeitamente aos pontos de treinamento, mas pode não capturar totalmente as amostras não utilizadas.

Logo, ambos os modelos confirmam que a saída do sistema é mais sensível a x_3. Treinar apenas com os pontos de treinamento ajusta os coeficientes para se adequar melhor a essas amostras, mantendo a capacidade de representação linear do sistema. O Adaline multivariado fornece uma aproximação eficaz, permitindo entender quantitativamente como cada entrada contribui para a saída do sistema.

5. CONCLUSÃO

Os exercícios realizados demonstraram a eficácia do modelo Adaline na modelagem de sistemas lineares, tanto univariados quanto multivariados (HAYKIN, 2009; SCHEINER; MURRAY, 2001). No exercício 1, o modelo Adaline foi capaz de aproximar com precisão a transformação de um sinal senoidal de entrada em sua saída correspondente, obtendo coeficientes que indicam o deslocamento e o ganho aplicados pelo sistema (RUMELHART; HINTON; WILLIAMS, 1986). A comparação entre as saídas previstas e as reais, em dados de teste não utilizados no treinamento, evidenciou boa

capacidade de generalização do modelo, confirmando que a relação entre entrada e saída pode ser representada linearmente (NISE, 2020).

No exercício 2, o modelo multivariado demonstrou que a saída do sistema é principalmente influenciada pelo sinal x_3, embora x_1 e x_2 também apresentem contribuições relevantes (SCICLUNA et al., 2020). Treinar o modelo com todas as amostras permitiu capturar a tendência geral do sistema, enquanto a utilização apenas dos pontos de treinamento ajustados evidenciou como os coeficientes podem ser refinados para se adequar a um subconjunto representativo de dados, embora com ligeira perda de generalização (HAYKIN, 2009).

Os resultados confirmam que o Adaline multivariado é uma ferramenta eficaz para aproximação linear de sistemas com múltiplas entradas, permitindo quantificar a contribuição individual de cada entrada e fornecer previsões confiáveis. Além disso, a análise gráfica demonstrou visualmente a aderência do modelo aos dados, facilitando a interpretação e validação da relação linear entre entradas e saída (SCHEINER; MURRAY, 2001). Em síntese, os exercícios reforçam que modelos lineares simples, como o Adaline, podem oferecer soluções precisas e interpretáveis para sistemas dinâmicos, especialmente quando a transformação de sinais é contínua e suavemente variável, servindo como base para análises mais complexas ou para o desenvolvimento de sistemas de controle (OPPENHEIM; WILLSKY; NAVA, 1996; NISE, 2020).

6. REFERÊNCIAS

1. HAYKIN, Simon. *Neural Networks and Learning Machines*. 3. ed. Upper Saddle River: Pearson, 2009.
2. RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagating errors. *Nature*, v. 323, p. 533–536, 1986.
3. NISE, Norman S. *Control Systems Engineering*. 9. ed. Hoboken: Wiley, 2020.
4. OPPENHEIM, Alan V.; WILLSKY, Alan S.; NAVA, S. Hamid. *Signals and Systems*. 2. ed. Upper Saddle River: Prentice Hall, 1996.
5. SCHEINER, B.; MURRAY, R. M. Linear Regression and Adaptive Linear Neurons: Theory and Applications. *IEEE Transactions on Neural Networks*, v. 12, n. 3, p. 543–552, 2001.
6. SCICLUNA, J.; et al. *Machine Learning in Python with scikit-learn*. Birmingham: Packt Publishing, 2020.